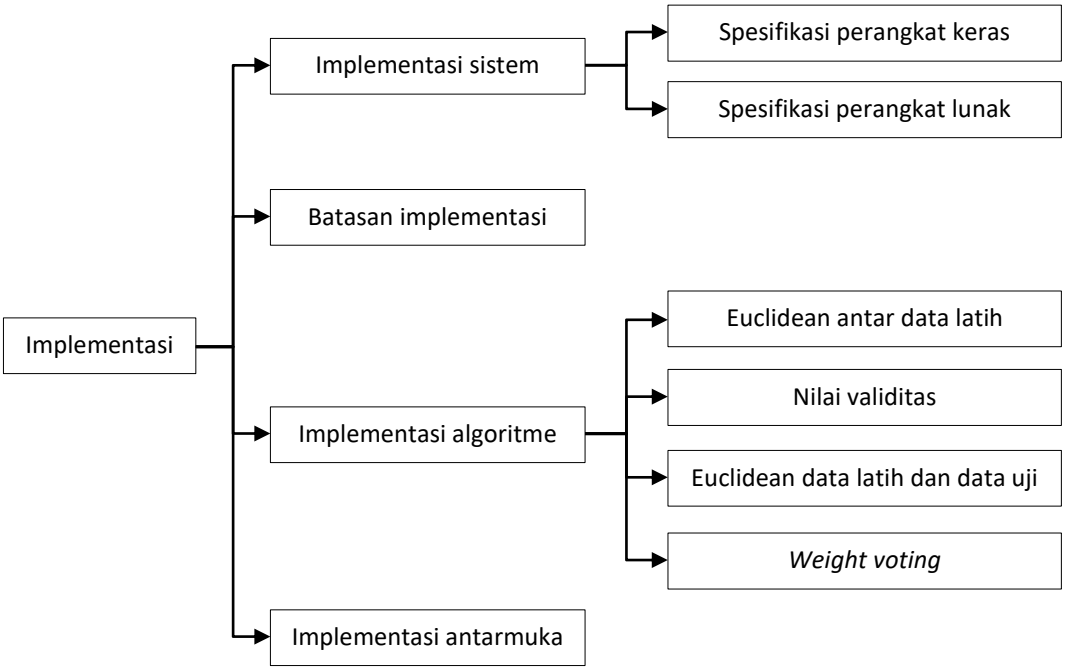
BAB 5 IMPLEMENTASI

Berdasarkan analisis kebutuhan dan perancangan yang telah dilakukan pada sebelumnya, bagian ini akan membahas tentang implementasi algoritme MK-NN untuk diagnosis penyakit anjing. Bahasan-bahasan dalam bab ini terdiri dari spesifikasi sistem, batasan implementasi, implementasi algoritme, dan implementasi antarmuka, yang ditunjukan pada Gambar 5.1.



Gambar 5.1 Bagan Implementasi

5.1 Implementasi sistem

Berdasarkan Gambar 5.1, implementasi sistem memerlukan dua aspek penting agar sistem dapat berjalan dengan baik sesuai fungsinya. Dua aspek tersebut adalah perangkat keras dan perangkat lunak yang digunakan untuk membangun sistem.

5.1.1 Spesifikasi Perangkat Keras

Tabel 5.1 menunjukan daftar spesifikasi perangkat keras yang digunakan untuk membangun sistem.

Tabel 5.1 Spesifikasi Perangkat Keras

Nama Komponen	Spesifikasi
Prosesor	Intel(R) Core(TM) i3-3217U CPU @ 1,80GHz
Memori	4 GB
Kartu grafis	Intel(R) HD Graphics 4000
Hardisk	500 GB

5.1.2 Spesifikasi Perangkat Lunak

Tabel 5.2 menunjukan daftar spesifikasi perangkat lunak yang digunakan untuk membangun sistem.

Tabel 5.2 Spesifikasi Perangkat Lunak

Nama	Spesifikasi
Sistem operasi	Windows 10
Bahasa pemrograman	Java
Tool pemrograman	Netbeans 8.0.1, JDK

5.2 Batasan Implementasi

Adapun batasan-batasan dalam implementasi algoritme MK-NN untuk diagnosis penyakit anjing, yaitu:

1. Sistem dibangun berbasis Java menggunakan bahasa pemrograman Java.
2. Masukan sistem berupa nilai bobot dari gejala-gejala penyakit dan diagnosis penyakit yang diberikan oleh dokter hewan. Masukan tersebut dibagi 2 menjadi data latih dan data uji.
3. Keluaran sistem berupa hasil perhitungan algoritme MK-NN, hasil klasifikasi penyakit anjing dan nilai akurasi.

5.3 Implementasi Algoritme

Implementasi algoritme MK-NN untuk diagnosis penyakit anjing terdiri dari perhitungan jarak Euclidean antar data latih, perhitungan nilai validitas, perhitungan jarak Euclidean antara data latih dengan data uji, dan *weight voting*.

5.3.1 Implementasi Perhitungan Jarak Euclidean Antar Data Latih

Langkah pertama dalam klasifikasi menggunakan MK-NN adalah menghitung jarak antar data latih yang ditunjukan pada Source Code 5.1.

```
1   public void Hitung_Euclidean_Data_Latih(int data_latih[][]){
2     this.euclidean_data_latih = new double [data_latih.length]
3     [data_latih.length-1];
4        for (int i=0; i < data_latih.length; i++){
5          int kolom = 0;
6          for (int j=0; j < data_latih.length; j++){
7            double total = 0.0;
8            if(i != j) {
9              for (int k=0; k < data_latih[0].length; k++){
10               total+=(Math.pow(data_latih[i][k]-
11               data_latih[j][k],2));
12     }this.euclidean_data_latih[i][kolom]= Math.sqrt(total);
```

```
13        kolom++;
14      }}}
15
16    public void Set_Kelas_Data_Latih(String kelas_dt_latih[]){
17      this.kelas_data_latih = new String[kelas_dt_latih.length]
18      [kelas_dt_latih.length-1];
19
20      for (int i=0; i < kelas_data_latih.length; i++){
21        int count = 0;
22        for (int j=0; j < kelas_dt_latih.length; j++){
23          if ( i != j){
24              this.kelas_data_latih[i][count]                    =
    kelas_dt_latih[j];
25              count++;
26         }}
27    }}
```

Source Code 5.1 Perhitungan Jarak Euclidean Antar Data Latih

Penjelasan Source Code 5.1 adalah sebagai berikut:

1. Baris 1 sampai 11 adalah kode untuk menghitung total jarak Euclidean antar data latih untuk setiap kolomnya.
2. Baris 12 adalah kode untuk mengakarkan total jarak Euclidean yang telah didapatkan sebelumnya.
3. Baris 16 sampai 27 adalah kode untuk set kelas dari data latih.

5.3.2 Implementasi Perhitungan Nilai Validitas

Sebelum melakukan perhitungan validitas, hasil perhitungan jarak Euclidean yang didapatkan diurutkan terlebih dahulu dari yang terkecil ke yang terbesar untuk mendapatkan tetangga terdekat. Setelah diurutkan, diambil jarak terkecil sejumlah nilai k yang dimasukan pengguna. Selanjutnya adalah membandingkan kelas setiap data latih dengan kelas dari tetangga terdekatnta. Jika kelasnya sama maka nilai kesamaannya 1 dan jika berbeda maka nilai kesamaannya 0. Hasilnya kemudian dijumlahkan dan dibagi dengan nilai k untuk mendapatkan nilai validitas. Implementasi algoritme perhitungan nilai validitas ditunjukan pada Source Code 5.2.

```
1   public void Short_Euclidean_Data_Latih(){
2    double temp =0.0;
3    String temp_="";
4    for (int i=0; i < this.euclidean_data_latih.length; i++){
5      for (int j=0; j < this.euclidean_data_latih[0].length;
```

```
6     j++){
7       for (int k=0; k < (this.euclidean_data_latih[0].
8       length-1); k++){
9         if (this.euclidean_data_latih[i][k] >
10        this.euclidean_data_latih[i][k+1]){
11          temp = this.euclidean_data_latih[i][k];
12          this.euclidean_data_latih[i][k] =
13            this.euclidean_data_latih[i][k+1];
14          this.euclidean_data_latih[i][k+1] = temp;
15          temp_ = this.kelas_data_latih[i][k];
16          this.kelas_data_latih[i][k] =
17            this.kelas_data_latih[i][k+1];
18          this.kelas_data_latih[i][k+1] = temp_;
19       }}}
20  }}
21
22  public void Ambil_Sebanyak_K(int K,
23  double euclidean_data_latih[][],
24  String kelas_data_latih[][]){
25    this.K_euclidean_data_latih =
26      new double[euclidean_data_latih.length][K];
27    this.K_kelas_data_latih     =
28       new String[kelas_data_latih.length][K];
29     for (int i=0; i < K_euclidean_data_latih.length; i++){
30       for (int j=0; j < K_euclidean_data_latih[0].length;
31       j++){
32         this.K_euclidean_data_latih[i][j] =
33           euclidean_data_latih[i][j];
34         this.K_kelas_data_latih[i][j] =
35           kelas_data_latih[i][j];
36  }}}
37
38  public void Hitung_Validity(int K,
39  String K_kelas_data_latih[][], String Kelas_data[]){
40    this.validity = new double[K_euclidean_data_latih.length];
41    for (int i=0; i < K_euclidean_data_latih.length; i++){
42      This.validity[i] = 0.0;
43      for  (int  j=0;  j  <  K_euclidean_data_latih[0].length;
    j++){
```

```
44        if (Kelas_data[i].equalsIgnoreCase
45        (K_kelas_data_latih[i][j])){
46           this.validity[i] +=1;
47        }}
48        this.validity[i] = this.validity[i]/K;
49  }}
```

Source Code 5.2 Perhitungan Nilai Validitas

Penjelasan Source Code 5.2 adalah sebagai berikut:

1. Baris 1 sampai 20 adalah kode untuk mengurutkan hasil perhitungan jarak Euclidean dari yang terkecil ke terbesar.
2. Baris 22 sampai 36 adalah kode untuk mengambil jarak Euclidean terkecil sejumlah nilai k yang diinputkan pengguna.
3. Baris 38 sampai 49 merupakan kode program untuk mendapatkan nilai validitas, di mana baris 44 sampai 47 merupakan kode program untuk memberikan nilai 1 apabila kelas data latih sama dengan kelas dari data dengan jarak Euclidean terkecil yang diambil. Sedangkan baris 48 merupakan kode untuk menghitung nilai validitas.

5.3.3 Implementasi Perhitungan Jarak Euclidean Data Latih dan Data Uji

Langkah berikutnya yaitu menghitung jarak Euclidean antara data latih dengan data uji yang ditunjukan pada Source Code 5.3.

```
1   public void Hitung_Euclidean_DataLatih_DataUji(
2   int data_latih[][], int data_uji[][]){
3      this.euclidean_data_uji =
4        new double[data_uji.length][data_latih.length];
5
6      for (int i=0; i < data_uji.length; i++){
7        for (int j=0; j < data_latih.length; j++){
8        double tot = 0.0;
9        for (int k=0; k < data_latih[0].length; k++){
10         tot += (Math.pow(data_uji[i][k]-data_latih[j][k],2));
11       } this.euclidean_data_uji[i][j] = Math.sqrt(tot);
12  }}}
```

Source Code 5.3 Perhitungan Jarak Euclidean Data Latih dan Data Uji

Penjelasan Source Code 5.3 adalah sebagai berikut:

1. Baris 1 sampai 10 adalah kode untuk menghitung total jarak Euclidean antara data latih dengan data uji untuk setiap kolomnya.
2. Baris 11 merupakan kode program untuk mengakarkan total jarak Euclidean yang telah didapatkan sebelumnya.

5.3.4 Implementasi Perhitungan *Weight Voting*

Langkah selanjutnya adalah perhitungan *weight voting* untuk mendapatkan kelas data uji. Perhitungan *weight voting* melibatkan nilai validitas dan jarak Euclidean data latih dengan data uji. Implementasi *weight voting* ditunjukan pada Source Code 5.4.

```
1  public void Weighted_Voting(double Validitas[]){
2    for (int i=0; i < this.euclidean_data_uji.length; i++){
3      for (int j=0; j < this.euclidean_data_uji[0].length;
4      j++){
5         This.euclidean_data_uji[i][j] = Validitas[j]*
6         (1./(this.euclidean_data_uji[i][j]+0.5));
7  }}}
8
9  public void setKelas_Data_Uji(String kelas_dt_latih[],
10 int jumlah_data_uji){
11   this.kelas_data_uji = new String[jumlah_data_uji]
12   [kelas_data_latih.length];
13
14   for (int i=0; i < kelas_data_uji.length; i++){
15     for (int j=0; j < kelas_data_uji[0].length; j++){
16       this.kelas_data_uji[i][j] = kelas_dt_latih[j];
17 }}}
```

Source Code 5.4 Perhitungan *Weight Voting*

Penjelasan Source Code 5.4 adalah sebagai berikut:

1. Baris 1 sampai 7 adalah kode untuk menghitung *weight voting* setiap data uji.
2. Baris 9 sampai 17 adalah kode untuk set kelas perhitungan *weight voting* berdasarkan kelas dari data latih.

5.3.5 Proses Klasifikasi

Langkah terakhir adalah melakukan klasifikasi data uji. Pada proses ini, nilai *weight voting* diurutkan dari yang terbesar ke terkecil. Kelas dari *weight voting* terbesar merupakan kelas dari data uji. Implementasi proses klasifikasi ditunjukan pada Source Code 5.5.

```
1  public void Short_Euclidean_DataLatih_Uji(){
2    double temp =0.0;
3    String temp_="";
4    for (int i=0; i < this.euclidean_data_uji.length; i++){
5      for (int j=0; j < this.euclidean_data_uji[0].length;
```

```
6     j++){
7       for (int k=0; k<(this.euclidean_data_uji[0].length-1);
8       k++){
9         if (this.euclidean_data_uji[i][k] <
10        this.euclidean_data_uji[i][k+1]){
11           temp = this.euclidean_data_uji[i][k];
12            this.euclidean_data_uji[i][k] =
13             this.euclidean_data_uji[i][k+1];
14            this.euclidean_data_uji[i][k+1]   = temp;
15
16            temp_ = this.kelas_data_uji[i][k];
17            this.kelas_data_uji[i][k] =
18             this.kelas_data_uji[i][k+1];
19            this.kelas_data_uji[i][k+1]   = temp_;
20      }}}
21 }}
22
23 public void Hasil_Klasifikasi(String kelas_data_uji[][]){
24    this.hasil_klasifikasi =
25      new String[kelas_data_uji.length];
26    for (int i=0; i < this.hasil_klasifikasi.length; i++){
27      this.hasil_klasifikasi[i] = kelas_data_uji[i][0];
28 }}
```

Source Code 5.5 Proses Klasifikasi

Penjelasan Source Code 5.5 adalah sebagai berikut:

1. Baris 1 sampai 21 adalah kode untuk mengurutkan nilai *weight voting* dari yang terbesar ke yang terkecil.
2. Baris 23 sampai 28 adalah kode untuk set kelas data uji berdasarkan kelas dari *weight voting*.

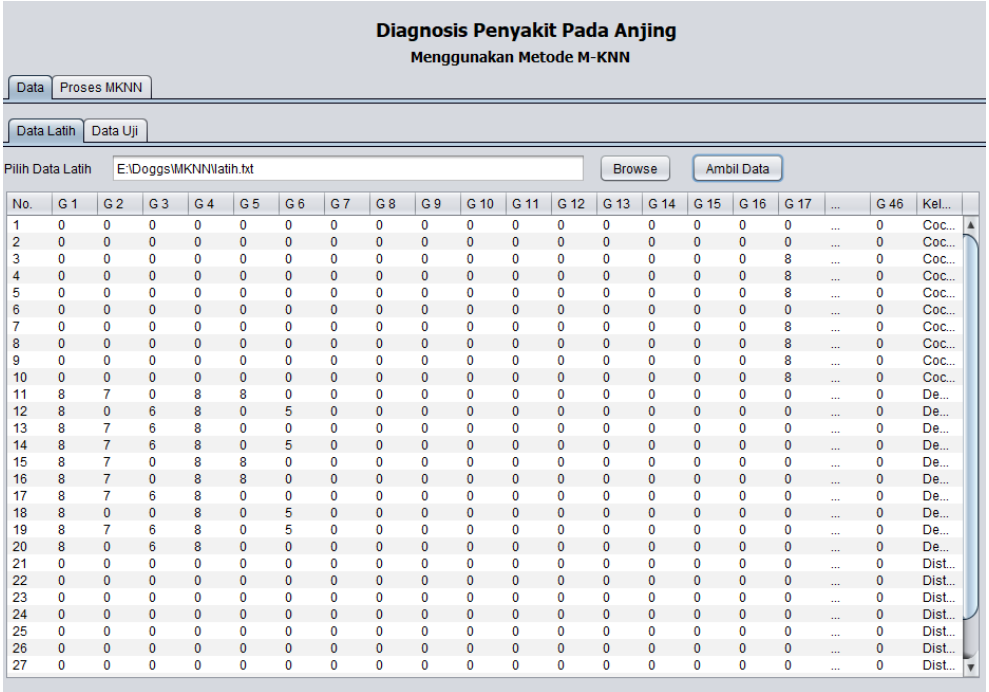
5.4 Implementasi Antarmuka

Antarmuka bertujuan untuk memudahkan interaksi pengguna dengan sistem, Implementasi antarmuka terdiri dari tampilan antarmuka awal, data uji, proses MK-NN.

5.4.1 Implementasi Antarmuka Awal

Tampilan awal merupakan tampilan pertama saat pengguna menjalankan aplikasi. Pada halaman awal terdapat judul aplikasi, menu Data, menu Proses MK-NN, menu Data Latih dan menu Data Uji. Saat aplikasi dijalankan akan

langsung menampilkan menu Data Latih yang berfungsi untuk memasukan data latih. Pengguna dapat memasukan data latih berupa file *notepad* yang isinya akan ditampilkan pada aplikasi. Implementasi antarmuka awal ditunjukan pada Gambar 5.2.



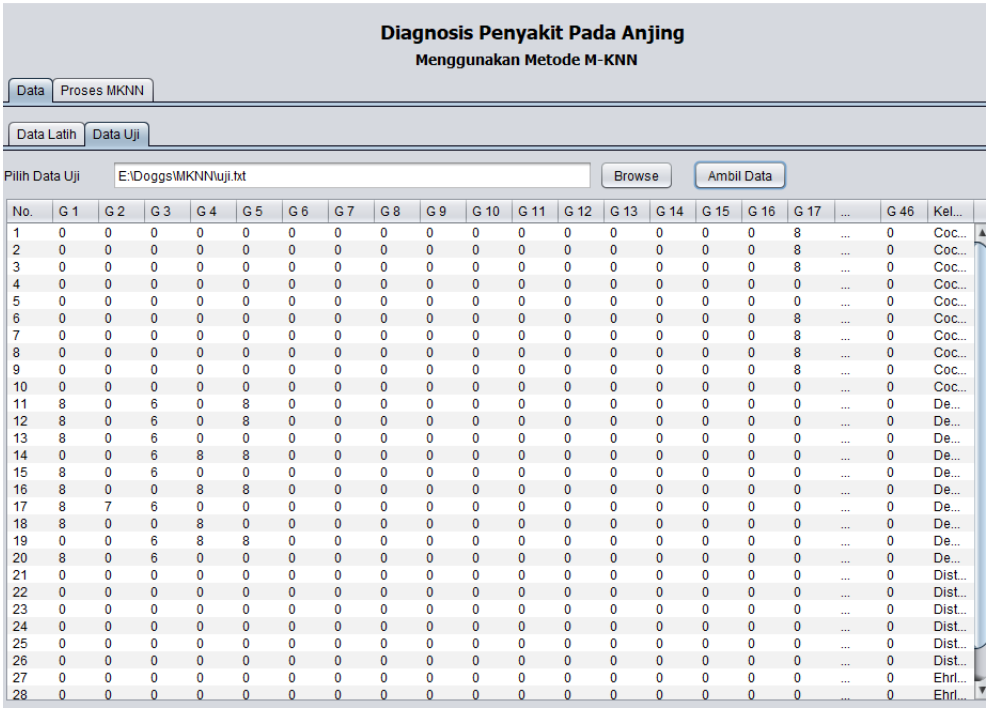
Gambar 5.2 Antarmuka Awal

5.4.2 Implementasi Antarmuka Data Uji

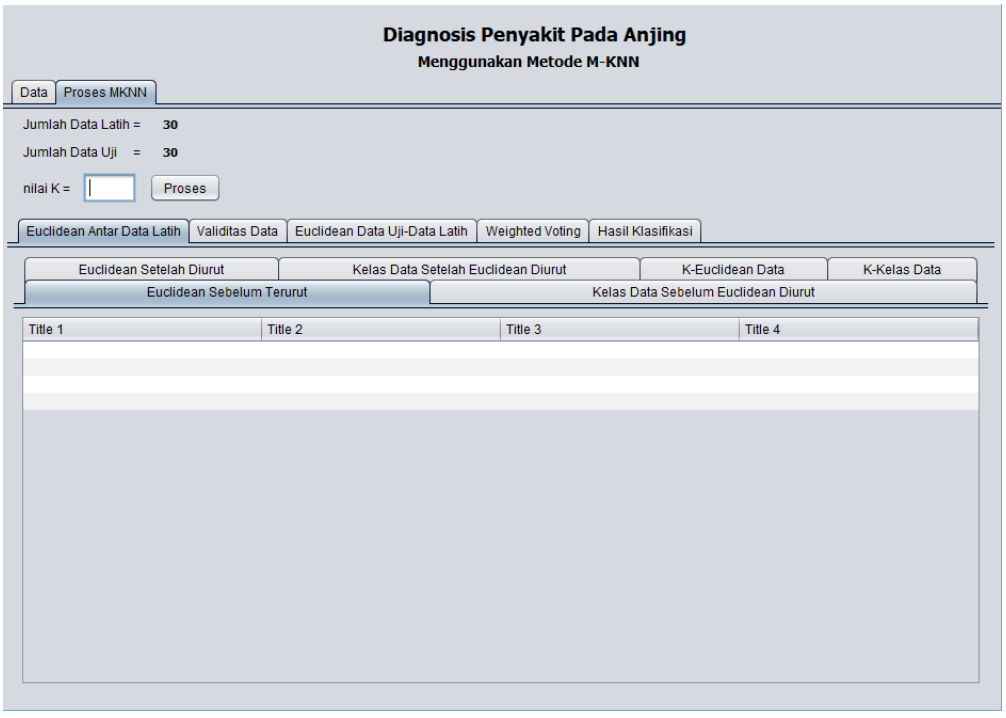
Menu Data Uji yang berfungsi untuk memasukan data uji. Pengguna dapat memasukan data uji berupa file *notepad* yang isinya akan ditampilkan pada aplikasi. Desain antarmuka tampilan data uji ditunjukan pada Gambar 5.3.

5.4.3 Implementasi Antarmuka Proses MK-NN

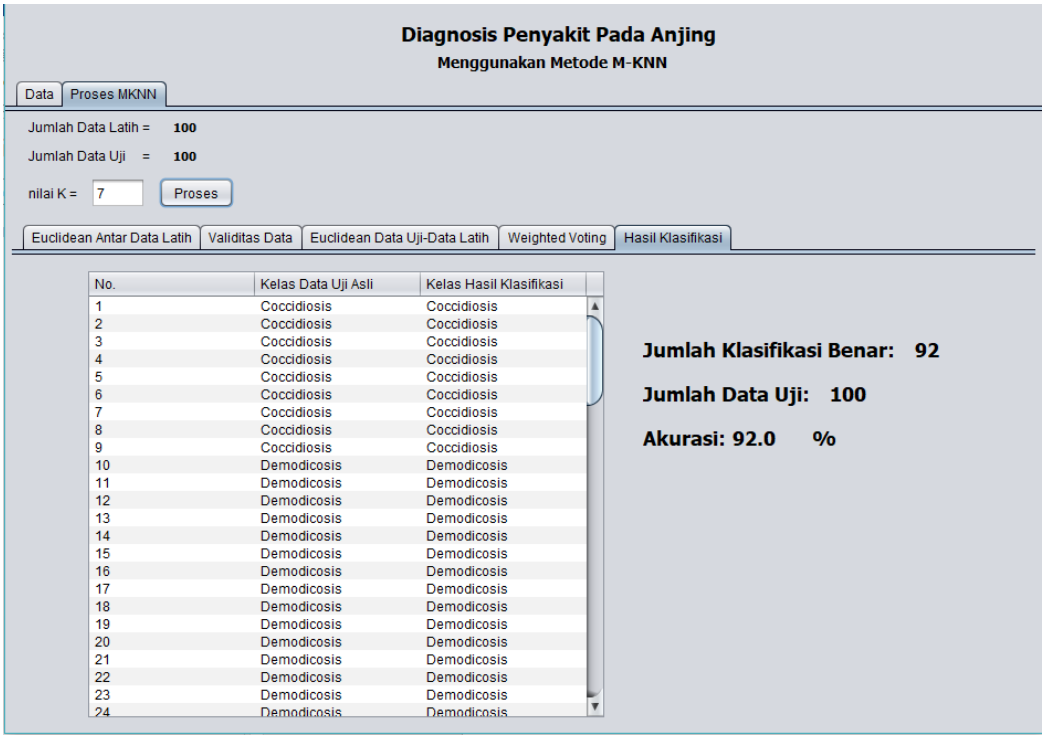
Menu Proses MK-NN merupakan menu untuk klasifikasi data uji yang diberikan. Pada menu ini, pengguna dapat menginputkan nilai k kemudian menekan *button* Proses untuk melakukan klasifikasi. Pada menu ini, pengguna dapat melihat hasil perhitungan setiap proses MK-NN. Tampilan menu Proses MK-NN sebelum dilakukan klasifikasi dapat dilihat pada Gambar 5.4. Sedangkan tampilan menu Proses MK-NN setelah dilakukan klasifikasi dapat dilihat pada Gambar 5.5.



Gambar 5.3 Antarmuka Data Uji



Gambar 5.4 Antarmuka Proses MK-NN Sebelum Dilakukan Klasifikasi



Gambar 5.5 Tampilan Proses MK-NN Setelah Dilakukan Klasifikasi